Release Notes
For
IEEE 1685™: IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

This document contains release notes for the 2022 revision of the standard collected and managed by the Accellera IP-XACT Working Group. IEEE 1685 can be downloaded here.

Removed features

- Conditionality of elements has been removed by removing the **isPresent** element in the whole schema. Backward compatibility of this feature is supported by means of Accellera Vendor Extensions. The Accellera IP-XACT Working Group provides a Vendor Extension schema and an XSL transform supporting up-conversion from ipxact:isPresent to accellera-cond:isPresent. The IP-XACT schema has been extended with vendor extension at certain locations to accommodate for these Accellera Vendor Extensions. IP-XACT design environments and tools do not have to support Accellera Vendor Extensions to be IP-XACT compliant.

Added and extended features

- **Short description and top-level displayName**
  Each element that can have a description, can also have a **shortDescription** element now. Also, top-level elements can have a **displayName** element. Short descriptions are more compact compared to descriptions. Display names can be used as identifiers outside of IP-XACT documents such as human-readable documents or graphical user interfaces.

- **Serial and multiplexed protocols**
  Abstraction definition ports have been extended with **packets**. A **packet** describes the organization of bits on the encapsulating logical port in terms of packet fields.

- **SystemVerilog interface connectivity and HDL struct and union connectivity**
  Component ports have been extended with **structured** ports, in addition to wire and transactional ports. A structured port describes the structure of an HDL struct or union or an SV interface. Its leaf nodes are wire ports that can be referenced in bus interface port maps.

- **Connectivity rules**
  Abstraction definition ports have been extended with element **match** with a boolean value indicating that the logical bits mapped onto this logical port in port maps shall match on all bus

interfaces of an interconnection. Also, attribute **allBits** with a boolean value has been added to the width element in wire ports to indicate that all logical bits shall be mapped in any port map.

- **Analog-/Mixed Signal**
  Component port descriptions have been extended with **domainTypes** and **signalTypes**. Domain types describe user-defined port disciplines. Signal types describe signal representations of ports using one of four enumeration values: continuous-conservative, continuous-non-conservative, discrete, and digital.

- **Power**
  Component **powerDomains** can be described and component ports can reference a power domain to indicate in which power domain the port is located. In hierarchical components, component instance power domains can be linked with hierarchical component power domains to describe power hierarchies.

- **Qualifiers**
  The set of qualifiers (**isClock**, **isReset**, **isAddress**, **isData**) has been extended. Also, qualifiers have been added to component ports, in addition to abstraction definition ports.

- **Port parameters**
  Parameters have been added to component ports which can be used in the same way as parameters in bus interfaces, registers, and register fields.

- **Module parameters for configuration at run-time**
  The enumeration values for attribute usageType has been extended with value **run-time**, in addition to values typed and nontyped. Also, the use of typed and nontyped has been changed to match with the proper terminology in VHDL and Verilog.

- **Memory-related parameterized type definitions**
  A new top-level element **typeDefinitions** has been added enabling definitions of
  - register field access policies (**fieldAccessPolicyDefinitions**),
  - register field enumerated values (**enumerationDefinitions**),
  - register fields (**fieldDefinitions**),
  - registers (**registerDefinitions**),
  - register files (**registerFileDefinitions**),
  - address blocks (**addressBlockDefinitions**),
  - banks (**bankDefinitions**),
  - memory maps (**memoryMapDefinitions**),
  - memory remap (**memoryRemapDefinitions**).

  A typeDefinitions element has formal modes, resetTypes, and views that are bound to actual modes, resetTypes, and views when the definitions are instantiated. A typeDefinitions element also has parameters and assertions. The parameters can be configured through configurableElements which can be propagated through design hierarchies. Type definitions may be instantiated in components or other type definitions.

- **Operating mode specific register access**
  A component **modes** element has been added which describes different operating modes of a component. A **mode** can have a **condition** to indicate when the mode is active. Such a condition is a boolean expression that can include terms for run-time values of port slices and register field slices and conditions of other modes in the same component. The modes element replaces the remapStates and alternateGroups elements. In addition, modes can be referenced to describe different access properties for address blocks, register files, registers, and fields. Also port slices and register field slices can be linked and these links can be mode specific. The access values of an element and its encapsulating elements are applied as filters to compute the actual access value. SCRs pertaining to access value restriction have changed accordingly. For instance, a register field with read-write access inside a register with read-only access results in read-only access for that register field.

- **Register file access**
  A registerFile can now also describe accessPolicies that contribute to computing access to the encapsulated bits**.**

- **No access and read response**
  The enumeration values for access have been extended with **no-access**, in addition to read-write, read-only, write-only, read-writeOnce, and writeOnce. For readable fields, the **readResponse** can be described using a constant value, for instance, to support “read-as-zero” or “read-as-ones”.

- **Field aliasing and broadcasting**
  Register fields can be **aliases** of other register fields. The behavior of aliased fields is as if the fields share the same physical storage bits. Also, register fields can **broadcast** to other register fields. The behavior of broadcast is that a value written to one field in a broadcast is written to all fields in a broadcast.

- **CPU memory map**
  In the cpu element, the addressSpaceRef element has been replaced by a **memoryMapRef** element. Also the concept of range, width, and segments have been added to the cpu element, where segments are called **regions** to avoid confusion. In this way, the size and regions of a cpu element can be documented and the referenced memory map can be used to define the content of the cpu memory map using address blocks, banks, and subspace maps.

- **Arrays and strides**
  Address blocks and register fields can be multi-dimensional **arrays** in the same way as register files and registers. All arrays can have a (one-dimensional) **stride** to describe the distance between two consecutive array elements.

- **Index variables**
  Array dimensions can have an **index** variable that can be referenced in displayName, shortDescription, and description element values using escape sequence $ipxact_index_value(<indexVar>). Documentation generators may process this escape sequence to document the

referenced index variable appropriately.

- **Access handles**
  The type of element pathSegment has been changed into string expression. The SV function $sformatf can be used to format HDL paths. New functions $sformatf and $ipxact_index_value(<indexVar>) have been added to the IP-XACT SV expression language that can be used in the string expression.

- **Expression**
  The IP-XACT SV expression language has been extended with the following functions:
  - $sformatf(), to format strings in the elements of type stringExpression,
  - $ipxact_absdefport_value(), to describe run-time values of abstraction definition ports in packet field value expressions,
  - $ipxact_field_value(), to describe run-time values of register fields in mode condition expressions,
  - $ipxact_index_value(), to describe values of index variables,
  - $ipxact_mode_condition(), to describe run-time values of mode conditions in mode condition expressions,
  - $ipxact_packetfield_value(), to describe run-time values of packet fields in packet field value expressions, and
  - $ipxact_port_value(), to describe run-time values of component ports in mode condition expressions.

  These functions are only allowed on specific elements as documented in the standard.

- **File types**
  File types have been added to support additional languages and additional versions of languages, specifically for SystemC, SystemVerilog, SystemRDL, VHDL, and Spice.

- **Choices**
  Choices element has been added in busDefinition, abstractionDefinition, design, designConfiguration.

- **TGI**
  In addition to SOAP, REST is supported as standard transport layer service. The TGI API calls have been refactored to provide consistent names. As a result, the API is not backwards compatible.

Changes & fixes

- **Revision of system map calculation**
  subspaceMap baseAddress changed to a signed value to ease system map mapping to HW IP.

- **Rename of elements**
  Renamed whiteboxElement to clearboxElement. Renamed master to initiator. Renamed slave to target.